|  |
| --- |
| [Type the company name] |
| PROGRAM 6: ACT Scores |
| C++ |

|  |
| --- |
| Brett Kriz  4/2/2014 |

// Brett Kriz

// Program 6 - ACT Scores

//

// Includes, just incase

#include <algorithm> // Algorithms >.>

#include <cmath> // Basic Math Operations

#include <cstdlib> // control of stdlib

#include <cstring> // Maniplulation of strings

#include <ctime> // Time stuff

#include <fstream> // File stream (I/O)

#include <iomanip> // Manipulation of I/O

#include <iostream> // Input\Output

#include <sstream> // String stream

#include <stdio.h> // Stuff For random numbers

#include <stdlib.h> // Standard Library

#include <string> // String Stuff

#include <time.h> // More time stuff

using namespace std;

// Standard...

// Early Proto

//void file\_read();

void sleep(unsigned int mseconds);

// Variables

const int ARRAY\_SIZE = 250;

struct Student{

int Date;

string DateSTR;

int ID;

int Score;

};

struct ACTdata{

//int Dates[ARRAY\_SIZE];

//string IDs[ARRAY\_SIZE];

//int Scores[ARRAY\_SIZE];

Student student[ARRAY\_SIZE];

int Size; // Single lol

};

// PROTOS

void setup();

bool commands(int ind, ACTdata& data); // Must be by ref for any function that needs it

int menu();

// functions of intrest

void TestsByDate(ACTdata& data);

void ListACTs(ACTdata data);

void AverageACT(ACTdata data);

void AddStudent(ACTdata& data);

void DeleteStudent(ACTdata& data);

void SortByDate(ACTdata& data);

void SortByScore(ACTdata& data);

// Other

void DisplayList(ACTdata data, int d = 180); // shhhhhh its a secret

string getDateSTR(int arg);

bool Read\_IN(ACTdata& data, string des = "actscores.txt"); // Read in

bool isDate(int arg);

string INTtoSTR(int a);

int addi();

void zi();

// Its been a while but this is just soo much easier

/\*--------------------------------------------------------

Function: main()

Disc: What? You know...

--------------------------------------------------------\*/

int main(){

// Main

setup(); // Yup

// Create Main Variables

ifstream IN;

int x = 0;

string des = "actscores.txt";

int size = 0;

ACTdata data; // Data Structure full of Student Structures

data.Size = 0;

system("title Program 6: BUT FIRST! Compairing RAW and Sorted List (By Date, Chronologic)");

// start functions

bool go = true;

go = Read\_IN(data);

// Avert Terror

if (!go){

//cout << "\n\n\t Program couldn't read in file, Aborting run\n\a";

//

cout << "\n\t Need file path! \nPLEASE ENTER: ";

cin.ignore(25, '\n'); // Just to reduce errors

getline(cin, des);// read

go = Read\_IN(data, des);

if (!go) {

cout << "\n\n\t Program couldn't read in file, Aborting run\n\a";

}

}

// First sort

SortByDate(data);

// Test code

system("color 0F");

DisplayList(data,0);

cout << "\n~~~~~~First Sort Check~~~~~~~~~~\n\n";

system("pause");

while(go){ // could already have failed so check

setup(); // Decor

go = commands(menu(), data); // function

if (go) {system("pause"); sleep(150);}

} // end menu loop

setup();

cout << "\n\n\t\t Bye!\n\n\n";

system("Pause");

return 0;

}

/\*--------------------------------------------------------

Function: Setup

Disc: Setup the window quick

See 'onStart'

--------------------------------------------------------\*/

void setup(){

// Setup

system ("Title Program 6 - ACT Scores");

system ("Color 0A");

system("cls");

//cout << "Console Application Template" << endl << endl;

}

/\*--------------------------------------------------------

Function: Sleep

Disc: Sleeps for x many seconds

--------------------------------------------------------\*/

void sleep(unsigned int mseconds){

// Sleep...

clock\_t goal = mseconds + clock();

while (goal > clock());

}

int menu(){ // Prompts user with menu and options

int ans;

int ops = 7;

do {

cout << "Input a command Number from the following!";

string w = " ===\t"; // str additive

// lol options menu

// addi()

cout << "\n "<< 1 <<w<<"List Tests By Date"

<< "\n "<< 2 <<w<<"List by ACT score (High to Low)"

<< "\n "<< 3 <<w<<"Average of ACT scores"

<< "\n "<< 4 <<w<<"Add Student"

<< "\n "<< 5 <<w<<"Delete Student"

<< "\n "<< 6 <<w<<"Display List, as is"

<< "\n "<< 7 <<w<<"Exit"

// Turns out, using addi() would make this count backwards :/

// Oh well!

<< "\n\t << <<";// END STR

ops = 7;

//zi();

cout << "\n\tEnter Choice: ";

cin >> ans;

}while(ans < 0 && ans > ops);

return ans;

}

int N = 0;

int addi(){

return N++;

}

void zi()

{ N = 1; }

/\*--------------------------------------------------------

Function: Commands

Disc: A convience function to use commands

--------------------------------------------------------\*/

bool commands(int ind, ACTdata& data){ // considers input

// Accept commands

bool stop = false;

string order = "", instr = "";

/\*

cout << endl << "Input a command (clear, thetime, stop): ";

cin >> instr;

order = instr.substr(0, instr.find\_first\_of(char(32)));

\*/

switch(ind){ // Add something after...

case 1: TestsByDate(data); break;

case 2: ListACTs(data); system("title Program 6: List by Score"); break;

case 3: AverageACT(data); break;

case 4: AddStudent(data); break;

case 5: DeleteStudent(data); break;

case 6: DisplayList(data); break;

case 7: stop = true; break;

default: cout << "\n\aWhat?\n"; break;

}// end case

cout << "\n [\n";

return !stop; // negate bool

}

void DisplayList(ACTdata data, int d){

// NO SORTING HERE!

int size = data.Size;

int i = 0;

while (i<data.Size){// note data.Size not size

Student z = data.student[i];

cout << "\n #" << i

<< "\n\tID: " << z.ID //str var

<< "\n\tScore: " << fixed << setw(3) << z.Score

<< "\n\tDate: " << z.DateSTR << endl;

i++;

size--;

// call sleep if needed

sleep(d); // milis

}

cout << endl;

}

void TestsByDate(ACTdata& data){

int x =0;

int DIM = data.Size;

SortByDate(data); // Sort before use

DisplayList(data); // Display

}

void SortByScore(ACTdata& data){ // @@@BYSCORE

int DIM = data.Size;

Student temp;

for (int end = DIM - 1; end >= 0; end--)

{

for (int count = 0; count < end; count++)

{

// Current isgreater above?

if (data.student[count+1].Score > data.student[count+0].Score)

{

// SWAP

temp = data.student[count];

data.student[count] = data.student[count + 1];

data.student[count + 1] = temp;

}

}

}

}

void ListACTs(ACTdata data){

int x = 0;

int DIM = data.Size;

system("color 8A");

SortByScore(data);

// Display (not DisplayList)

while (x<DIM){

Student z = data.student[x];

cout << "\n#" << x

<< "\n\tID: " << z.ID // str

<< "\n\tScore: " << z.Score;

x++;

sleep(200);

// Just fo decoration man, thats it and thats all maaaan

}

system("title Program 6: Woo-WOOOOOO0");

}// end

void AverageACT(ACTdata data){

int x = 0;

int DIM = data.Size;

double avg = 0.0;

cout << endl << " {";

while (x<DIM){ // Sum all

int cur = data.student[x].Score;

cout << cur << " "; // Output numbers above

avg += cur; // increment x2

x++;

}

// Then Divide by total

avg = double(avg/DIM); // Multiply by 100

cout << "}\n";

cout << "The Average ACT score was: " << fixed << setw(5) << setprecision(2) << avg

<< " based on " << fixed << x << " students.\n";

}

string INTtoSTR(int a){

char buff [33] = "";

\_itoa\_s(a, buff, 10); // have to return the buffer

// Look into triming

return buff;

}

void AddStudent(ACTdata& data){

// get vars, slip in at end, sort

system("cls");

system("title Program 6: Add Student");

cout << "Add sudent:\n";

Student z;

bool b1, b2;

// ENTER ID

unsigned int preID = 0;

do{

cout << "\nEnter ID: ";

cin >> preID;

// bad id if

b1 = (preID < 1000 || preID > 9999);

//cout << b1;

if (b1) {cout << "\t Try agian, must be 4 numbers with atleast a least a leading 1";}

sleep(150);

}while(b1);

z.ID = int(preID);

// ENTER SCORE

do{

cout << "\nEnter Score: ";

cin >> z.Score;

b2 = (z.Score < 0 || z.Score > 32);

//cout << b2;

if (b2) {cout << "\t Try agian, must be between 1 & 32";}

sleep(150);

}while(b2);

// ENTER TEST DATE

do{

cout << "\nEnter Test Date (yyyyMMdd): ";

cin >> z.Date;

sleep(150);

}while(!isDate(z.Date));

// create string quick

z.DateSTR = getDateSTR(z.Date);

cout << z.DateSTR << "\n";

// Insert into array

data.student[data.Size] = z;// Add 1 to size and add new student

data.Size++;

SortByDate(data);

}// end func

string getDateSTR(int arg){ // GET DATE STRING

// create string quick

//check if valid;

// Credits to stackoverflow.com

string temp = INTtoSTR(arg); // <----

int ENDofYEAR = 4;

int ENDofMONTH = ENDofYEAR + 3;

temp = temp.insert(ENDofYEAR,"/"); // insert at end of 4 chars

temp = temp.insert(ENDofMONTH,"/"); // insert at end of MM

return temp;

}

bool isDate(int arg){

bool ans = (arg >= 19990101); // 1999 01 01?

if (!ans) {cout << "\n\t Year Must be before 1999\n"; return false;}

// Sice is a semi valid date...

int ENDofYEAR = 4;

int ENDofMONTH = ENDofYEAR + 3;

string z = getDateSTR(arg);

int mm = atoi(z.substr(ENDofYEAR+1,2).c\_str()); // x = atoi(string.c\_str())

int dd = atoi(z.substr(ENDofMONTH+1,2).c\_str()); // string > C string > Int

ans = ans && (mm<=12 && mm>0) && (dd > 0 && dd <= 31);

if (!ans) {cout << "\nMM = " << mm << " \t| DD = " << dd << "\n"; return false;}

// yyyy MM DD

// 8 long or past a certian date lol

return ans;

}

void DeleteStudent(ACTdata& data){

int x = 0, target=-1;

int DIM = data.Size;

bool waiting = true;

system("title Program 6: Remove Student");

do{

// Draw some stuff

system("cls");

system("color 07");

DisplayList(data);

// Request the file number

cout << " ^^^ \n"

<< " FILE # \n";

cout << "\tSelect a student by file number:";

cin >> target;

if (target >= 0 && target < DIM){

waiting = false;

// good data

}else{

system("color 04");

cout << "\nBad Input! Please retry!\n";

system("pause");

}

}while (waiting); // DO untill good IN

// Overwrite1

//cout << target << endl;

for(x = target; x<DIM-1; x++){

data.student[x] = data.student[x+1]; // drop it on it

}

data.Size--; // reduce size

// END stuff

system("cls");

SortByDate(data);

DisplayList(data);

cout << "\n Student Removed: #" << target << " : Original: " << DIM << " New: " << data.Size << endl;

}

void SortByDate(ACTdata& data){

int x=0,y=0;

int DIM = data.Size;

Student temp;

// Thanks for the bubble sort code Tim!

for (int end = DIM - 1; end >= 0; end--)

{

for (int count = 0; count < end; count++)

{

// Current isgreater above?

if (data.student[count].Date > data.student[count+1].Date)

{

temp = data.student[count];

data.student[count] = data.student[count + 1];

data.student[count + 1] = temp;

}

}

}// End Sort

}// END FUNC

bool Read\_IN(ACTdata& data, string des){

ifstream IN;

int x = 0;

bool good = true;

IN.open(des); // Modularity >:3

//cout << endl;

if (IN.is\_open()){

system("color 09");

while(!IN.eof()){

//IN >> data.Dates[x] >> data.IDs[x] >> data.Scores[x];

Student z;

int id; // going to str soon

IN >> z.Date;

IN >> id;

z.ID = id;

IN >> z.Score;

z.DateSTR = getDateSTR(z.Date);

data.student[x] = z;

// Something for checks

cout << "\n#" << x

<< "\n\tID: " << z.ID //str var

<< "\n\tScore: " << fixed << setw(3) << z.Score

<< "\n\tDate: " << z.DateSTR << endl;

x++; // yup increment

} // end loop

cout << "\nRead in complete!\n\n";

data.Size = x; // Store size

system("pause");

system("cls");

system("color 0A");

//size = x;

// not open?

}else{

cout << "\n\nInput File is empty!\n\t Check path!\a\n";

//cout << des << endl; //This wont work... idk

// '<<' : no operator found which takes a right-hand operand of type 'std::string' (or there is no acceptable conversion)

system("color CA");

system("pause");

system("color 0A");

good = false;

}// end if

IN.close();// Better close that...

return good;

}// end func

Output:   




